Interlingua based Neural Machine Translation

Carlos Escolano Peinado

Supervisors:

Marta Ruiz Costa-Jussà

Lluis Padró

June 27, 2018

Abstract

Machine translation is a highly interesting research topic with a big impact in multilingual information society. For the last 30 years, most state-of-the-art approaches have mainly been designed and trained for language pairs, translating from source to target languages. The number of translation systems required with these approaches scales quadratically with the number of languages which becomes very expensive in highly multilingual environments.

Current Neural Machine Translation approach has an architecture of encoder-decoder which inherently creates an intermediate representation. However, systems are still trained by language pair. In this thesis, we propose a new Neural Machine Translation architecture which aims at generalizing this intermediate representation to an interlingua representation shared across all languages on the system. We define this interlingua as the shared space in which the sentences can be represented independently of their language. This interlingua representation can be then generated and interpreted by all systems.

We propose a way of training such systems based on optimizing the distance between the representations generated and the use of variational and not variational autoencoders. Experimental results show that this technique can produce comparable results to the state-of-the-art baseline systems while giving clear evidences of learning the interlingua representation. To evaluate the quality of this interlingua representation, we use a new evaluation metric capable of measuring the similarity among the recovery of sentences from the corresponding interlingua representations and originated from different languages encodings of parallel sentences.

Acknowledgement

I would like to thank Marta Ruiz Costa-Jussà and Jose Adrián Rodriguez Fonollosa for their comments and guidance during this project. To my workmates that have witnessed this process and also help every time they could. And to my friends and family for their support these months.

Contents

1 Introduction

Machine translation has become a usual service in our lives thanks to services such as *Google Translate*. Traditionally those systems where either based on rules generated by linguists experts in the languages to translate or based on the probabilities calculated from a corpus of parallels sentences.

In recent years the state of the art for this task is Neural Machine Translation, in which a neural network is trained to learn the transformation of a source sentence in an intermediate representation that is later decoded into the target language. Even though these architectures have been proven to perform better that the previous paradigms they still present the limitation of requiring a great number of parallel examples in order to train a functional model.

1.1 Motivation

Current machine translation systems are centered in using a corpus of parallel sentences between two languages train a system that is able to learn how to translate from one language, the source language, to another, the target language. In case we want to be able to translate in both direction we would need to train two different systems, with the cost of time and resources that these processes require.

Now imagine that we have more than two language, n language, in order to translate from any language to any of the other we would need to train $(n-1)^2$ different translation systems each one requiring its own training process. Also at deployment all systems should be ready at all times waiting to be required, which also implies a cost in maintenance and hardware of those systems.

Another aspect in which nowadays machine translation architectures are not suitable is low resources languages. These are languages that due to have a small community of speakers have only small data sets of parallel data available to train a machine translation system. Additionally this data is usually available between the low resource language and English. In order to train a system to any other language an intermediate translation in English would be required.

1.2 Problem Statement

Imagine that different language could be represented in the same way, that the meaning of a sentence could be represented in the same way for different languages and could be understood by systems for each of those languages. This share representation is the Interlingua, a semantic representation that is valid for all the languages in the system.

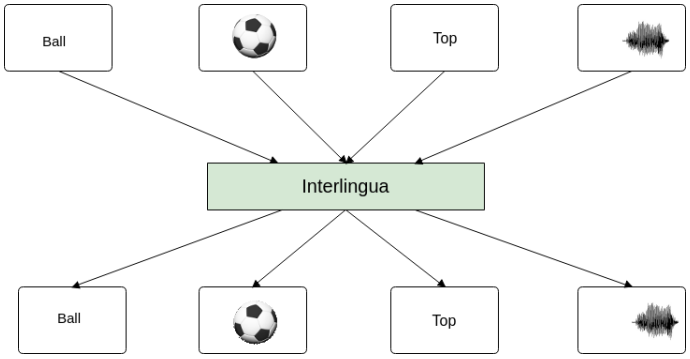

Fig. 1.1: Example of multimodal interlingua system

Figure 1.1 shows the objective of this interlingua, not only for text but for all data sources, If this theoretical system could be implemented a single system would be able to perform several tasks such as machine translation, image captioning in several languages, generate an image from its description and speech generation.

In this paper we are going to only work with one of those cases, machine translation

1.3 Contributions of the thesis

This thesis proposes a proof of concept for a neural machine translation model based on interlingua and autoencoders. The main advantages of the proposed architecture are:

- The number of systems required for the translation grows linearly with the number of languages. In opposition to the standard NMT systems in which each translation direction has to be trained individually.

- The proposed architecture trains all translation directions at once. In order to force all modules of the system to produce a compatible latent representation all modules are jointly trained.

1.4 Thesis Structure

The following paper will be organized in 8 main chapters.

Chapter 2: Theoretical background of the concepts that we are going to use in the proposed models

Chapter 3: State of the art. Discussing the current state of machine translation and how it related with the topic of this work.

Chapter 4: Proposed architecture. Discussing the several alternatives proposed.

Chapter 5: Toy problem. A small problem to illustrate the principle applied in a smaller task.

Chapter 6: Results. Experiments conducted during this work and model performance discussion.

Chapter 7: Visualization. Visualization of the latent space.

Chapter 8: Conclusions. Discussing the achieved aspects of this project and future work.

2 Theoretical background

In this chapter we are going to discuss the theoretical aspects needed for the proposed architectures. Three main techniques where employed for this development, Transformer Architecture, Autoencoders and Canonical Correlation Analysis.

2.1 Transformer

Previous neural machine translation models where based on the use of Recurrent Neural networks to keep information about the context of the sentence as the different tokens where processed. Transformer[Vas+17], which is the current state of the art, is based on applying self attention to both the input and the output of the network.

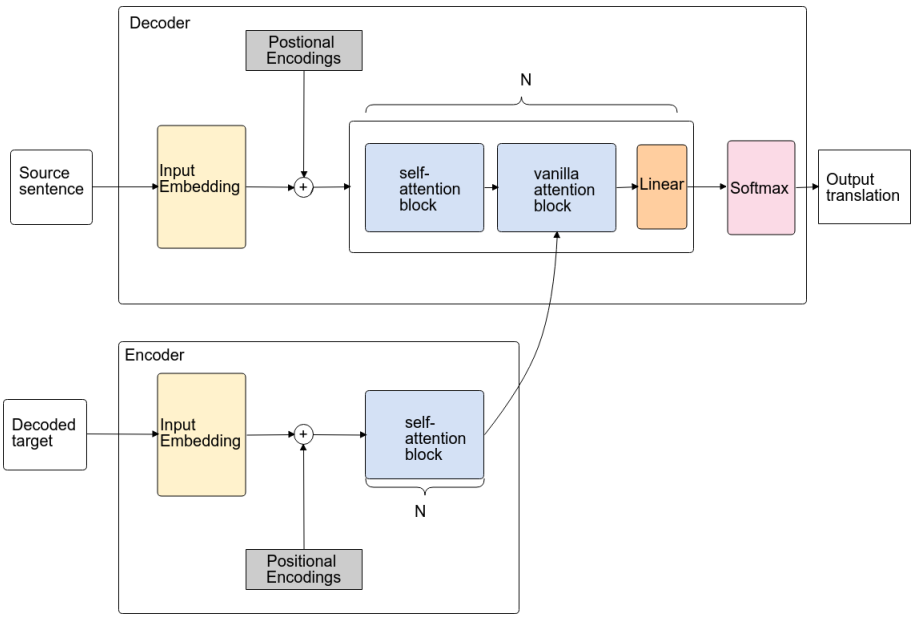

Fig. 2.1: Transformer overview

Figure 2.1 shows the full architecture of a transformer model. Essentially it follows the traditional Encoder-Decoder architecture in which the the input sentences are first encoded in a fixed sized continuous representation to therefore be decoded in the target sentence. The main idea of this architecture is applying self-attention

repeatedly over the transformed data to both encode and decode the final representation.

The first step of the model to feed data to the system is the embedding layer. This is a usual embedding layer as previously employed in Sequence to Sequence models in which for each token in the sentence will be represented as a vector of fixed size, transforming an input of size $d \in \mathbb{N}^l$ into a continuous representation $enc \in \mathbb{R}^{e \cdot l}$. Where l is the maximum number of tokens in the sentence and e is the size of the embedding representation.

A disadvantage of this architecture compared to previous Sequence to Sequence [BCB14] architectures is that the full sentence is treated at once and no context is kept of the previous tokens in the sentence. In order to add this information in the sentence representation a positional encoding is added to the resultant embedding of the previous layer. Two possible alternatives can be employed for this task, a sinusoidal function that provides a different value for each position of the sentence, or a learned transformation. The sinusoidal function even though has proven to generalize better in case of using sentences of varying length during test.

The sum of the embedding and positional encoding are then feed into the attention blocks as seen in figure 2.2. These blocks consist in several sub-blocks called heads where a fragment of the input representation is processed. Each heads input consists in the three following components:

1. **Query:** These values are the previous information over which calculate the attention matrix that would be applied over the values.

2. **Keys:** Similar to a dictionary this input acts as the different values to calculate the attention matrix.

3. **Values:** The data over which the attention matrix would be applied.

The final attention is calculated using the following expression:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{(d_k)}})V \tag{2.1}$$

Where d_k is the length of the input. This acts as an scale factor for the final value of the attention matrix. Figure2.2 show this process in a block diagram. It show an addition step of masking that diagonalizes the matrix before the softmax that outputs the final attention matrix. The reason for this step is that during inference it

prevents the attention mechanism to attend at the part of the matrix that has not been generated jet and only contains padding.

After each attention block its output is added to the input of the block in a process called Residual Connection. Finally layer normalization[BKH16] is performed. This normalization consists in computing a mean and variance to normalize the summed output of the neurons of the layer. By applying this process the authors claim that the training process is faster and it provides improvements for convergence.

During the encoding process this attention block is repeated n times each of them using as input the output of the previous attention block. The output of the final block of the encoder is the encoded continuous representation of the source sentence that would be later use for the decoding process.

The decoding part of the architecture follows a similar structure. First and embedding layer transform the discrete representation of the tokens of the target sentence into a continuous fixed size vector representation. During training the decoder would use the full target sentence as input while during inference it would get a vector containing only the token employed for padding and as each token is being decoded if would be added to this vector until the end of sentence. The reason for this substitution is that for the decoding of a token the model needs the encoded representation provided by the encoder and also the previous tokens generated, in a similar way as a recurrent network decoder would do.

The first step of the decoder analogously to the encoder would perform self-attention over the decoder input by applying an attention block. This block is followed by a block of attention over the encoding of the source sentence. This block is also called *vanilla attention* because it is more similar to the traditional concept of attention in Sequence to Sequence models in which the attention is performed in the decoder over the steps of the recurrent encoder's output.

In *vanilla attention* the values and the keys are the encoding of the source sentence and the query is the output of the decoder self attention block. The resultant attention matrix will be calculated from the context of the already decoded output and then applied to the encoded source sentence to generate the next token.

The final step of the model is applying a linear transformation that outputs as many values as possible tokens in the target dictionary and a softmax layer to show pick the most probable token.

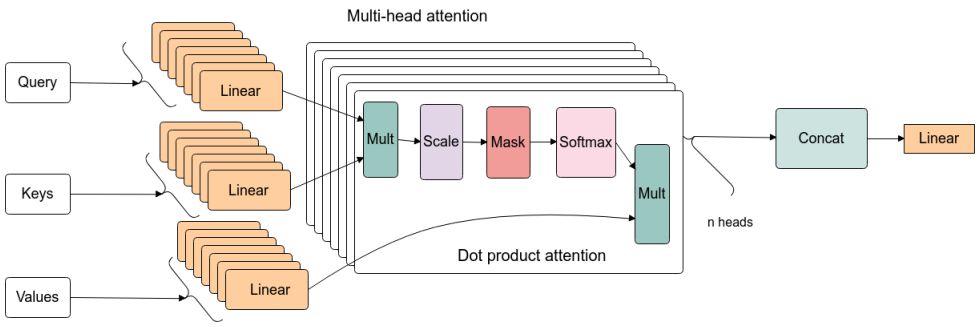

Fig. 2.2: Transformer multihead attention. Original image from [Vas+17]

2.2 Autoencoders

Autoencoders can be defined as networks that are able to learn from input data x a function $h(x)$ that is able to reproduce x through an internal latent representation. They are considered an unsupervised method as no labeling or extra information is required, due to their objective of reconstructing input data. This process can be divided in to steps:

- **Encoding**: In this step a space transformation is calculated from input data. Figure 2.3 shows in blue the encoder part of the network which learns a transformation of the data in a lower dimensionality space.

- **Decoding**: Process of transforming the latent representation back into the input data.

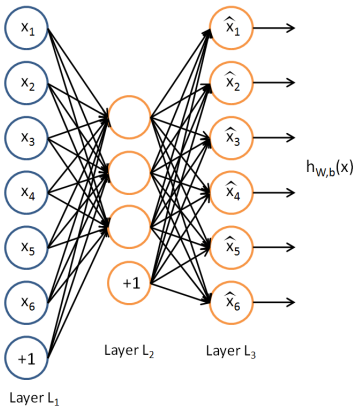

Fig. 2.3: Example of autoencoder architecture

The rationale behind these models is that this latent encoding provides us with a representation that can be employed for several tasks. Some of the most common uses are, serve as feature for other models and reduction dimensionality, by reducing

the dimensionality of the internal representation we are effectively mapping our data in a space that from which it can be recovered back.

In order to train the system both encoder and decoder part are jointly trained and optimized based on a reconstruction loss. This loss measures the differences between the input data and the generated reconstruction. It is also important to notice that only enforcing the network to reproduce its output could lead to learning an identity function without generalization, An example of this could be a model with to many units per layer that is able to represent each training example as a unique number that the decode can recover, without learning any relation between data. To prevent it some techniques such as denoising autoencoders [Vin+08], that add some noise to the input data that the network has to learn to remove, or sparse autoencoders [Ng11], where the neuron activation is limited by design to force sparcity in the representation, can be employed.

We have talked about their capabilities for data representation but they also can be used as generative data of new cases, specially variational autoencoders. Given that our decoder has learn how to transform a latent representation into the original input data, if a new encoding is generated from any other source, for example a probabilistic distribution, new examples never seen in our data could be generated that could be similar to the ones seen.

2.2.1 Variational autoencoders

Following the case of autoencoders as a generative models, any new encoding could be generated without actual data, but the results would depend of the distribution employed. Variational autoencoders [KW13] follow the same encoder-decoder structure of general autoencoders but instead of learning a representation of the examples in the training data they learn the parameters of the distribution which encodings will be sampled from.

Given $P(x)$ the distribution of our data and $P(z)$ the distribution of the latent representation we want to train, we can express $P(X)$ as:

$$P(X) = P(X|z)P(z) \tag{2.2}$$

Meaning that the data distribution can be expressed as the conditional probability of $P(x)$ given a latent representation z times the probability of z $P(z)$. The idea is to find as an optimization problem the distribution $P(z|X)$ that can recover the data.

For example, given a training set we could choose a Guassian distribution as the distribution of our latent representation. The model would optimize the μ and σ so that the data can be correctly recovered. Ones the model is trained we could randomly sample from that distribution new latent representations z' that are not originated from actual data but the decoder is able to use to generate similar examples to our data.

2.2.2 Vector quantization

A particular case of variational autoencoders is Vector Quantization Variational Autoencoders (VQ-VAE) [O+17]. In this technique instead of optimizing the parameters of a probability distribution from which the encodings can be sampled from what the network is defining is a table that defines all the possible values that the latent representation can take.

We define a table T of size $n \cdot d$ where n is the number of different vectors in the table and d is the dimensionality of each of those vectors. In order to choose the right vector from the table the closest vector in the table is retrieved following the expression:

$$z_q(x) = argmin(|z_i(x) - x|) \tag{2.3}$$

We can imagine this algorithm as a clustering algorithm in which each vector of the table corresponds to a centroid identifying a class from the input. During each training batch centroids are optimized to better represent the information of the sentences belonging to that class and during inference the closet vector is retrieved.

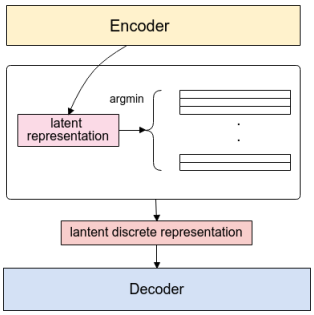

Fig. 2.4: VQ-VAE architecture

2.3 Canonical correlation analysis

Canonical Correlation Analysis(CCA)[Hot35] measures the the relationship of two variables. It produces new bases that equal or smaller in dimensionality to the smallest variable that we want to finds its correlation. It can be defined as finding two vectors X and Y which correlation between their projections can be mutually maximized.

Chandar et al. (2015) focus on the objective of common representation learning. This work it is inspired by the classical Canonical Correlation Analysis [**hotelling:1936**] and it proposes to use an autoencoder that uses the correlation information to learn the intermediate representation. In this paper, we use this correlation information to measure the distance among intermediate representations following the expression:

$$c(h(X), h(Y)) = \frac{\sum_{i=1}^{n}(h(x_i - \overline{h(X)}))(h(y_i - \overline{h(Y)}))}{\sqrt{\sum_{i}^{n}(h(x_i) - \overline{h(X)})^2 \sum_{i}^{n}(h(y_i) - \overline{h(Y)})^2}} \tag{2.4}$$

Where X and Y are the data sources we are trying to represent, $h(x_i)$ and $h(y_i)$ are the intermediate representations learned by the network for a given observation and $\overline{h(X)}$ and $\overline{h(Y)}$ are the mean intermediate representation for X and Y, respectively.

3 State of the art

The work presented in this paper is based in two fundamental research topics: machine translation and interlingua representation. In this section we are going to briefly discuss the previous work in these areas and their current state of the art.

3.1 Machine translation

Machine translation has been a research topic for a long time due to its many applications and theoretical interest in the area of text representation and understanding. Until few years ago translation systems were based on rules generated by experts in the languages that manually generated rules and grammars that correctly depict the differences between those languages [Arn94].

Statistical machine translation changed the state of the art due to its ability to learn a translation between a source and target language based on the token distribution in a given parallel corpus. These models are based on generating a language model to compute the conditional probability given the previous tokens in the sentence. A variation of this principle is the phrase-based machine translation in which next token is not generated using the previous token but the n previous tokens in the sentence. *Moses* [Koe+07] is the standard software employed and it is still in use as baseline and in some task where few parallel corpus is available.

It was in 2014 when first results [BCB14] showed that neural networks could outperform statistical models in translation. Previous models where based on a encoder-decoder architecture where the source sentence is encoded into a fixed size vector representation from which later using recurrent decoder target tokens are generated. Those vanilla sequence to sequence models present some drawbacks such as that they can get stuck generating a token that can be repeated several times in the decoded sentence. Also in addition to over translate a segment of the sentence parts of the source sentence could not be translated at all. To prevent these situations attention mechanisms [LPM15] were added. Attention mechanism consist in an additional layer that computes the the attention that each part of the fixed size sentence encoding and a softmax function to normalize the outcome. In [BCB14], was proposed a bidirectional encoder that consists in two recurrent layers one that

read the sentence forward and one that reads it backwards and how the decoder reads the context to generate a token.

Neural machine translation using recurrent encoder-decoder architectures where the state of the art until 2017 where to different models where presented. First, in [Geh+17] where a convolutional model with attention was proposed and outperformed recurrent models to that day with the same test data. On the other hand in [Vas+17] the Transformer architecture was proposed where the encoding and decoding are entirely based on self-attention over the data.

3.2 Interlingua representation

In this thesis we are defining interlingua a shared latent space where sentences are represented by the network encoders. These sentences represented in the space can also be retrieved into the original sentence or a translation by the decoder.

Several work has been conducted in this direction. One of the first successful attempts was [Joh+16]. In this work they trained a single encoder-decoder network where all languages where fed and could produce all of the trained languages. In order to disambiguate the the decoding language a special token was added at the beginning of the sentence to indicate the target language. This model followed the recurrent sequence to sequence architecture presented before. In their work they proved that the system was able to translate language pairs that where never seen together in the training process.

More recent works such as [Sch+17] avoid the idea of a single network that has to learn all the languages and that could be difficult to train and focus on architectures where each source and target data have their own encoder decoder sharing a common intermediate layer to try to force the representation to lay on the same space. This models present and advantage over the previous one network architectures because they are easier to employ in multimodal tasks. As each information source only depends of their own encoding data such as text or image can be easily added without sharing weights and modifying the encoder for the needs of each source.

Also in the same line of work in [Lu+18] they present a model in which they train a set of language specific encoders and decoders with parallel data only between pairs. Their model also relied on recurrent sequence to sequence models with the addition of a shared recurrent layer with attention to transform language dependent encodings into language independent ones. To achieve they propose a different training schedule where for each batch only one encoder and decoder was trained

from a pool of system including the autoencoder ones and the systems to and from English that acted as a pivot language during the training schedule.

Another area where shared spatial representation is commonly used is image captioning. This task consists in given an image the network has to generate a description of the image. First works such as [Vin+15] and [KF15] consist in the use of image embeddings that are later used as initial context vector of a recurrent encoder-decoder.

In the recent work [Gel+17] they present a model that joins both concepts of generation of a interlingua representation between different network applied to image captioning. In this work a convolutional model is trained to generate a representation o a given image and then one or more recurrent encoders, each one for a different language. The main idea is train the encoders to produce a representation as closer to the image as possible and there close between them. They claim that their model outperforms previous image description systems.

4 Architecture

4.1 Architecture overview

Given a parallel corpus our objective is training an encoder and decoder for each of the languages that are compatibles with the other components through a common intermediate representation generated by both encoders and understood by both decoders. For this, we propose a novel architecture and within it, we experiment with different distance measures and both discrete and continuous intermediate representations.

The architecture consists in an autoencoder for each of the languages to translate. Each network consists in a transformer network [Vas+17]. The advantage of using this model instead of other alternatives such as recurrent or convolutional encoders is that this model is based only on self attention and traditional attention over the whole representation created by the encoder. This allows us to easily employ the different components of the networks (encoder and decoder) as modules that during inference can be used with other parts of the network without the need of previous step information as seen in Figure 4.1.

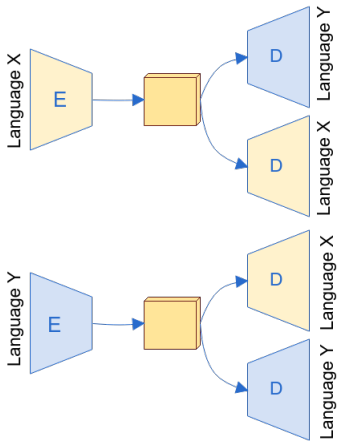

Fig. 4.1: Architecture diagram. Every module is compatible with the intermediate representation.

As follows, we detail this architecture step-by-step for a particular example. Imagine that we have a pretrained English-Turkish model using this architecture. This model

consists in two autoencoders, one for English and one for Turkish, which at the end of the day, it means that we have two encoders (one for each language) and two decoders (again, one for each language).

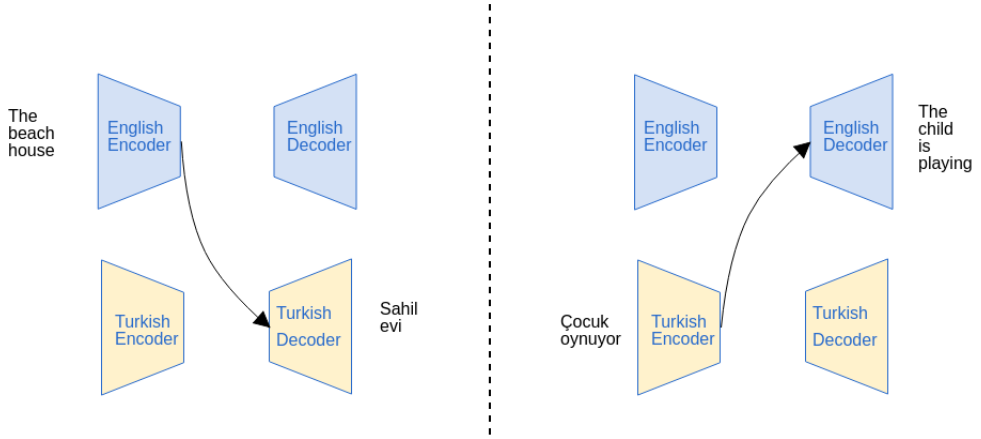

Fig. 4.2: Example of English to Turkish and Turkish to English translation

Given the English sentence *The beach house* and the Turkish sentence *Çocuk oynuyor* the model will translate them as shown in Figure **??**. For English-to-Turkish translation, the English sentence is fed to the English encoder (from the English autoencoder) which generates a fixed-size representation. This representation if used with the English decoder would just reconstruct the input. Given that the internal representation is shared between the two autoencoders, by using the Turkish decoder (from the Turkish autoencoder), we can generate a translation into this language. In this case, from the English sentence *The beach house*, the system will be able to output the Turkish sentence *Sahil evi*. Using the same trained model we can also generate the Turkish-to-English translation. To translate the Turkish sentence *Çocuk oynuyor*, we feed the sentence into the Turkish encoder (from the Turkish autoencoder) and use the encoded sentence as input to the English decoder (from the English autoendoer) to generate the translation.

4.2 Discrete latent representation

The internal representation of the Transformer architecture [Vas+17] is a continuous two dimensional representation of the input sentence $e \in \mathbb{R}^{nd}$ where n is the maximum number of tokens in the sentence and d is the embedding size employed. This high dimensional matrix representation depends on the previous self-attention layers. Given the high dimensionality of the internal representation, it is challenging to define a procedure to compute distances among different representations.

In this paper we propose an alternative for this representation by using variational autoencoders. In previous section 2.2.2, we have presented the idea of vector quantization and how it can be employed in autoencoders. By having all possible codifications in the table we have only to optimize that the right vector is selected, instead of generating the right point in the multidimensional space. Also this table can be shared across the encoders in the system, which ensures that all of them can produce the same exact vectors for the same parallel sentence in all languages.

Previous work [KB18] has shown that for the task of translation this technique presents a problem that the authors call "riches get richer" where due to the limited number of possible representations during training only a few vectors get most of the representations and therefore the network is not paying attention to the encoding and it is focusing its generation on the decoder input.

To prevent this situation a variation of this techniques was implemented: Decomposed Vector Quantization [KB18]. In this technique instead of having a single vector table that contains all the possible latent representations, we split the table in k tables, each of which only encode $\frac{d}{k}$ values of the vector. The final latent representation is the concatenation of the vectors retrieved from each table. This provides the advantage that instead of n possible encodings (being n the size of the table), n^K encodings can be represented using the same memory space (if having k tables).

It is also worth noticing that this vector quantization approach is not differentiable. To apply backpropagation to the network, the gradient flow from the decoder to the encoder has to be manually handled. For each encoder-decoder in the system its reconstruction loss can be defined as the combination of the three following components:

- **Decoder loss:** Cross-entropy loss between the generated tokens and the target sentence. This loss is the same loss employed in the not variational autoencoder case.

- **Decomposed vector quantization loss:** This step is not differentiable. Therefore gradient flow to the encoder is stopped in this layer. This loss term is calculated as the squared difference between the generated encoding and the vector quantization vector.

- **Encoder loss:** Similar to the previous term but in this case, the gradient flow is stopped in the decomposed vector quantization layer.

4.3 Optimization

In order to achieve the desired universal language that can be used by all the modules of the system, all the components have to be optimized simultaneously. This is a difference to traditional neural machine translation systems in which translation is only considered between the source and target language. In the proposed architecture, all languages are equally considered and both translation directions are generated during the training process. To achieve it, we design the following loss function:

$$Loss = L_{XX} + L_{YY} + L_{XY} + L_{YX} + d(h(X), h(Y)) \tag{4.1}$$

Where L_{XX} is the autoencoding loss for language X, L_{YY} is the autoencoding loss of language Y, L_{XY} is the translation loss from language X to language Y and L_{YX} is the translation loss from language Y to language X.

The final term of the loss is the measure of the distance between the two intermediate representations $h(X)$ and $h(Y)$ of both autoencoders. For this distance, we propose:

1. *Correlation distance* which measures how correlated are the intermediate representations of the autoencoders for each batch of the training process:

$$d(h(X), h(Y)) = 1 - c(h(X), h(Y)) \tag{4.2}$$

2. *Maximum distance* which measures the closeness of the intermediate representations as the maximum of the difference of the representation of a source and its target sentence:

$$d(h(X), h(Y)) = max(|h(X) - h(Y)|) \tag{4.3}$$

5 Toy example

In this chapter we are going to discuss a toy problem over image data in order to visually exemplify the objectives of the proposed training methodology.

5.1 Problem description

Having parallel image data, in this example handwritten numbers. We create an artificiall parallel data set in which for each image we generate a new transformed image that has been transposed and color inverted. The aim of this example is to prove that it is possible to train two autoencoders on the parallel data. These autoencoders are able to learn a common representation with only the shared distance-based loss as source of information.

To test the model's performance we will try to encode an image using one of the encoders, either the one that has only seen original images or the one that has only seen the transformed ones. And feed the encoding to the other autoencoder's decoder. If learned representation is shared and provides meaningful information for the task we would either obtain a transformed version of the image or reverse the transformation.

As dataset we use MNIST [LCB10] which is a database of handwritten numbers in gray-scale. In this particular implementation we are using the data packed with the standard *Tensorflow* library which consists in 60.000 training images and 10.000 test images.

5.2 Network architecture

In this simplified example we are going to define a reduced network architecture and a new simpler distance measure to use as distance term of the loss.

The employed network consists of an autoencoder with a single fully connected layer of 300 units and linear activation and output layer of 784 units with also linear activation. This way the hidden layer produces a coding of 300 dimensions from the

784 input dimensionality of the flattened MNIST images that will be decoded back into the original dimensionality by the output layer.

The loss function is based on three main terms, the reconstruction loss of the first autoencoder, the reconstruction loss of the second autoencoder and the distance loss. By reconstruction loss we define the mean square difference of the input image and the generated image of the autoencoder following the expression:

$$loss = rec_loss(x_1, h_1(x_1)) + rec_loss(x_2, h_2(x_2)) + dist_loss(h_1(x_1), h_2(x_2)) \tag{5.1}$$

$$rec_loss(x, h(x)) = squarex - h(x) \tag{5.2}$$

$$dist_loss(h_1(x_1), h_2(x_2)) = \square(x - y)^2 \tag{5.3}$$

The network has been trained for 10.000 updates with batch size 100. As optimizer was used Adam[KB14] with the default parameters, learning rate equals 0.001, beta1 equals 0.9, beta2 equals 0.999 and epsilon equals 1e-08

5.3 Experimental results

Once the networks are trained we can analyze the reconstructed images. Given the latent representation generated from one of the encoders we can try to decode it using both decoders and compare their results. In figure5.1 we can see how both reconstructions correctly represent the characteristics of the number such as the line overlap on top of the 0 or how the 6 in the last image is not closed. It is also important to notice that the cross decoded images present noisier edges than the original ones, but mostly the important information about the number is preserved.

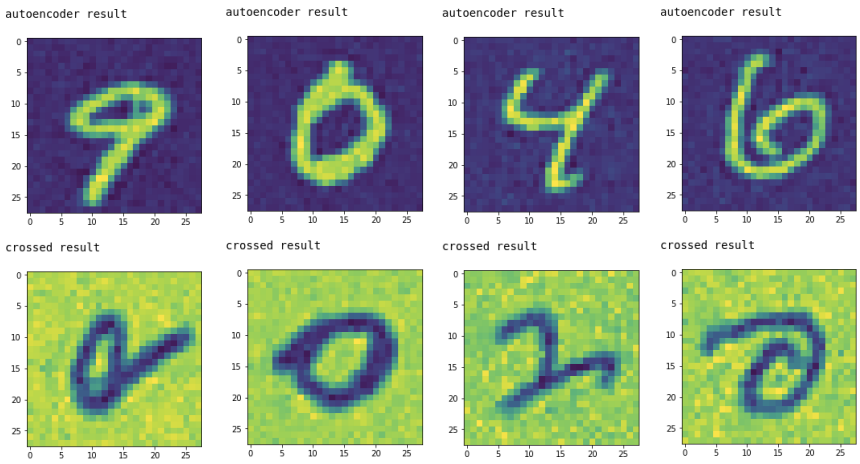

Fig. 5.1: Examples of generated outputs. Upper row autoencoder generated. Bottom row cross generated.

6 Experiments

In this chapter, we report details on the experimental framework and results. We include a brief description of the evaluation measure, the dataset, the hyper parameters of the system and finally, the results.

6.1 Evaluation measure

To evaluate the quality of the proposed translations in this work, we are going to use the standard metric BLEU[Pap+02]. This measure is the most widely used metric in the area. BLEU consists in measuring the accuracy of the proposed translation for different sizes of n-grams, generally from 1 to 4, using the expression:

$$Pn = \frac{\sum_{C \in candidates} \sum_{n-gram \in C} count(n-gram)}{\sum_{C' \in references} \sum_{n-gram' \in C'} count(n-gram')} \tag{6.1}$$

Generating the right n-grams is important for the translation because it means that the right words in the right order are being generated by the system. A problem than can arise from this measure is that we can create a system that correctly generates the first n-grams of the sentence but then it stops. This candidate sentence even producing the right n-grams is missing the meaning of the sentence by not fully translating it. The following brevity penalty is introduced to prevent this undesired behavior:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1-\frac{r}{c}} & \end{cases} \tag{6.2}$$

If the candidate sentence c is longer that the reference r no penalty is applied but, if not, its value in the metric is reduced. Also for each n-gram size a weight is calculated. In the general case, we can assume $w_n = \frac{1}{N}$ where N is the number of n-gram sizes evaluated. Finally, the BLEU metric is calculated as:

$$BLEU = BP\dot{e}xp(\sum_{n=1}^{N} w_n \dot{l}og(Pn)) \tag{6.3}$$

Measuring the distance between the intermediate representations produced by the networks depends on the relative distance of the vectors in those representations for the sentences in their own language. To overcome this difficulty, we propose a novel approach to measure their quality based on the performance of the models for the autoencoding and translation tasks.

The intuition behind this concept is that, independently of the quality of the translation, an effective universal language should produce close results when a decoder is fed with an intermediate representation created by the encoder of the same language or a different language. We propose to measure this difference with the *BLEU* score of both outputs using the autoencoder output as the reference to compare with the translation output.

6.2 Data

The dataset employed for these experiments is the SETIMES2 dataset consisting in sentences extracted from parallel news in Turkish and English. To prepare the data to be used with the model several preprocessing steps were required. We employed the following pipeline using standard scripts from the statistical machine translation tool *Moses*[Koe+07]:

1. Elimination of empty sentences. Empty lines in addition to not provide useful information for training could raise errors when training the *Moses* baseline.

2. Tokenization. In order to create a vocabulary that represents the occurrences of words in the training set is important to have a consistent separation of tokens in the corpus. For example, splitting words from punctuation marks prevents the system from identifying those words as new words with low frequency instead of two already known tokens.

3. Cleaning. Sentences length may affect the performance of the system. In this step sentences longer than 50 tokens are removed from the training corpus. In case just one of the two parallel sentences is too long both are deleted.

4. True Casing. In languages were there are differences between upper and lower case is important to correctly capitalize words in the same way for training

and test sets. In this step a model is trained to learn which words have to be capitalized in addition to beginning of sentences and words that appear just after a period or punctuation marks.

Table6.2 shows the number of sentences, words and number of unique words in the corpus. It is a small dataset with approximately 200.000 sentences but Turkish is not a language in which a lot of parallel corpus can be found and having a small dataset allows us to perform more experiments. Training a competitive machine translation system for high resource language pair such as German-English could take several days to train.

Language	Set	Sentences	Words	Vocabulary
Turkish	Train	200290	4248508	158276
	Dev	1001	16954	6463
	Test	3000	54128	15898
English	Train	299290	4713025	73906
	Dev	1001	22136	4318
	Test	3000	66394	9503

Tab. 6.1: Corpus Statistics. Number of sentences (S),words (W), vocabulary (V)

6.3 Hyper parameters

In this section, we report the hyper parameters used for each architecture proposed. All experiments have been implemented using *Tensorflow* using a *NVIDIA TITAN* gpu with 12 GB of RAM memory.

6.3.1 Autoencoder architecture

Translation models usually take days to train and perform so due to hardware and time constraints limited parameter tuning could be performed on the models. Most of the following parameters are the chosen parameters in the original paper [Vas+17]. The principal changes are the hidden representation size, that has been reduced from 512 to 128 in order to fit both autoencoders in a single gpu and batch size that has been also reduced due to gpu memory constraints. The full list of parameters is shown in tabke 6.2

Hparam	Value
Embedding size	128
Hidden units	128
Training steps	30000
Encoder attention blocks	6
Decoder attention blocks	6
Attention heads	8
batch size	32
maximum sentence length	50
optimizer	Adam
learning rate	0.0001

Tab. 6.2: Autoencoder architecture hyper parameters

As stopping criterion we choose to finish the training process if translation loss was not decreasing for 1000 batches sampled every 100 batches. Figures 6.2 and 6.1 show how loss converges for the different network scenarios. It is worth noticing how both accuracy and loss start to converge faster in the autoencoder task than in the translation task. But at the end of the training process they arrive to similar values of over 90% accuracy for the training data.

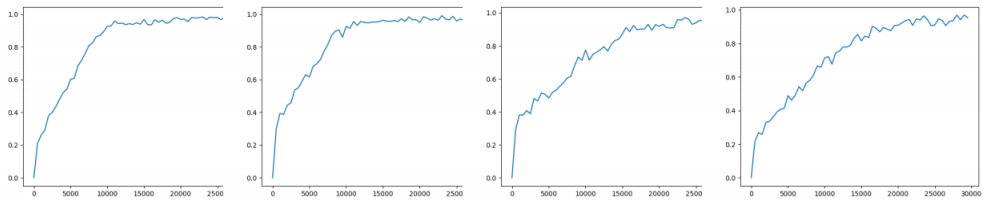

Fig. 6.1: Accuracy through the training process for the English autoencoder, Turkish autoencoder, English-Turkish translation and Turkish-English translation repectively.

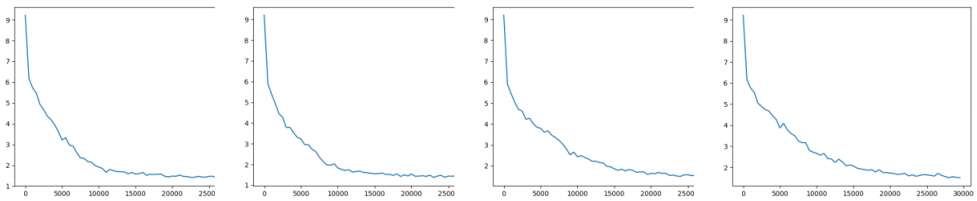

Fig. 6.2: Loss through the training process for the English autoencoder, Turkish autoencoder, English-Turkish translation and Turkish-English translation repectively.

6.3.2 Decomposed Vector Quantization architecture

For this alternative most of the parameters are chosen equally that the non variational model due to the same limitations of time and gpu memory. The *Decomposed Vector*

Quantization specific parameters of this model, number of tables and tables size are maintained from [KB18]. Table 6.3

Tab. 6.3: Decomposed Vector Quantization hyper parameters

Hparam	**Value**
Embedding size	128
Hidden units	128
Training steps	90000
vq tables	4
table size	5000
Encoder attention blocks	6
Decoder attention blocks	6
Attention heads	8
batch size	32
maximum sentence length	50
optimizer	Adam
learning rate	0.0001

It is surprising that this alternative took 3 times more to converge than the non variational alternative and that the final accuracy of the model for the training set was approximately of 75% compared to the 90% of the other model.

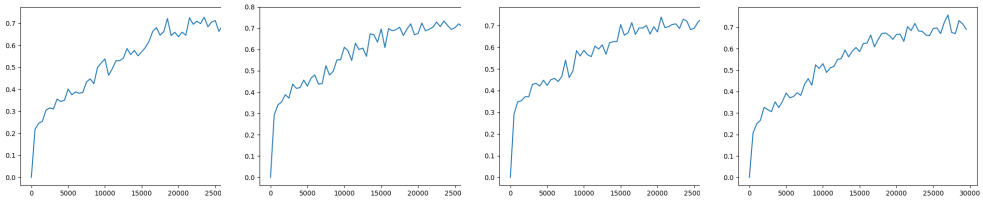

Fig. 6.3: Accuracy through the training process for the English autoencoder, Turkish autoencoder, English-Turkish translation and Turkish-English translation repectively.

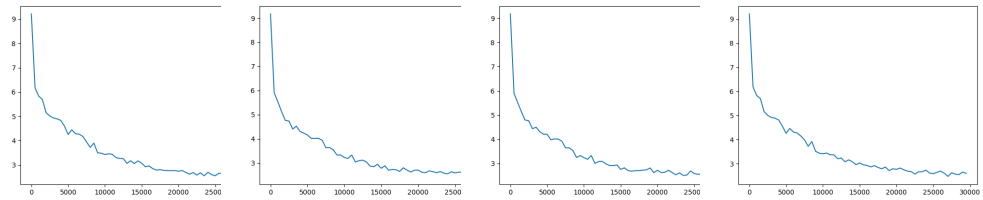

Fig. 6.4: Loss through the training process for the English autoencoder, Turkish autoencoder, English-Turkish translation and Turkish-English translation repectively.

6.4 Results

In order to measure the performance of the proposed architecture we are going to compare its performance to the following baselines:

- **Moses**: This statistical machine translation model was trained using
- **Character-based neural machine translation:** For the embedding of the source sentence, we use set of convolutional layers which number kernels are (200-200-250-250-300-300-300- 300) and their lengths are (1-2-3-4-5-6-7-8) respectively. Additionally 4 highway layers are employed. And a bidirectional LSTM layer of 512 units for encoding. The maximum source sentence's length is 450 during training and 500 for decoding both during training and inference.
- **Default transformer:** A default transformer model trained to translate from a source to a target language with the default optimization and the same parameters as the non variational autoencoder version of the proposed architecture.
- **DVQ transformer:** A transformer model using decomposed vector quantization as latent representation. Trained with the same parameters as the variational autoencoder alternative of the proposed architecture.

Tab. 6.4: BLEU results for the different system alternatives, Moses, RNN, Transformer baseline, and different configurations of our architecture, Universal (Univ) with and without decomposed vector quantization (dvq), and correlation distance (corr) and maximum of difference (max)

	EN-TR	**TR-EN**
Moses	7.25	11.10
RNN Char/Char	5.87	6.23
Transformer	8.32	12.03
Transformer dvq	2.89	8.14
Univ + corr	8.11	12.00
Univ + max	6.19	10.38
Univ + dvq + corr	7.45	7.56
Univ + dvq + max	2.40	5.24

After evaluating how the proposed systems perform compared to the baseline systems, we can also evaluate how the internal representation affects the text generated. To do so we are going to evaluate the results of each decoder in the three following scenarios:

- **Autoencoder:** Each decoder's input is the latent representation produced by the encoder of the same language.

- **Translation:** Each decoder's input is the latent representation produced by the encoder or the other language.

- **BLEU between Autoencoder and Translation:** How the performance changes from using the decoder as autoencoder or as translation. In the ideal case, autoencoder and translation should provide the same results and therefore 100 BLEU would be obtained.

These results are shown in Table 6.5 only for the best performing model from Table 6.4, the non variational autoencoder model with correlation distance, performs in autoencoding and translation. A big performance gap can be observed between the autoencoder and translation results.

Tab. 6.5: Comparison of BLEU scores on the *univ+corr* architecture when performing as autoencoder and translation (MT). The third column (A-T) is the BLEU between autoencoder and translation outputs.

Decoder	**Autoencoder**	**MT**	**A-T**
EN	63.32	12.00	11.90
TR	59.33	8.11	6.02

Finally, Table 6.6 show some examples of generated output by the systems in both directions

Tab. 6.6: Example of generated sentences in both directions English-Turkish and Turkish-English respectively

	Language	Example
Input	English	in 911 Call , Professor Admits to Shooting girlfriend
Target	Turkish	Profesör 911 ' i arayarak Kız arkadaşını öldürdüğünü İtiraf Etti
Output	Output	Profesör 9 ' i arayarak Kız İtiraf Etti
Input	Turkish	bunlar artık Lamb ' in yanıtlayamayacağı sorular .
Target	English	those are questions lamb can no longer answer .
Output	Output	a question lamb cannot answer.

7 Interlingua-visualization

In order to fully understand how data is represented in the latent space and the relation of sentence pairs, we developed a REST API version of the model and the server infrastructure to settle a visualization tool. This tool consists in a preprocessing step that queries the server until n sentences representations are downloaded and then, UMAP[MH18] is applied as reduction of dimensionality technique to plot the sentences in 2D [Aje18].

Figure 7.3 shows the plot of the latent representation of 100 parallel sentences obtained from the best performing model from Table 6.5, which is the non variational architecture with correlation distance.

A clear separation can be observed between the English (green) sentences and Turkish (yellow) sentences. Even though points are close in the space, having two different clusters means that there are some transformations that are not correctly reduced by the objective distance that we are using. As a consequence, it is the decoder who has to remove this extra noise from the encoding in order to use the representation as context for the translation. These two different clusters that we have in the representation may also cause the BLEU differences between the autoencoder and the translation generation.

Fig. 7.1: Overview of the interlingua space. Showing 100 English(green) sentences and their Turkish translations(yellow)

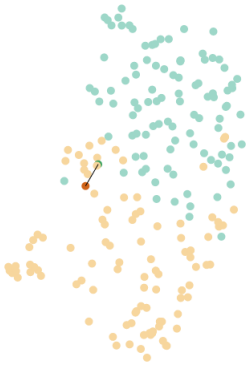

Fig. 7.2: Example of close sentences in the space. Showing 100 english(green) sentences and their Turkish translations(yellow)

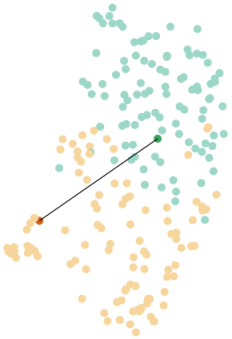

Fig. 7.3: Example of far sentences in the space. Showing 100 english(green) sentences and their Turkish translations(yellow)

8 Conclusion

This thesis has proposed and developed a new neural machine translation architecture capable of learning an interlingua representation for different languages. Our experimental results show that our proposed variational architecture not only shows evidences of learning an interlingua representation, but it also achieves comparable results to current state-of-the-art in translation and it is also able to outperform some other strong models. Our variational autoencoder architecture is specially designed to scale to more languages and multimodal tasks. Additionally, this thesis has proposed an evaluation metric for translation which is able to evaluate the quality of the interlingua representation.

Even though our proposed system is able to translate using only the interlingua representation (without the source information), there is still room for improvement in unifying the intermediate representation of different languages. These results are coherent with the fact that there is still a gap in the performance between the autoencoder and translation results. To solve this gap, it may be required to define a most suitable distance loss or a post process that allows us to learn a transformation between representations.

Additional further work towards improving the architecture include testing several aspects like the scalabilty of the architecture and how add new languages to a pretrained architecture. From another side and based on the explained MNIST toy problem, it would be interesting to test the architecture in multimodal tasks such as image captioning or even speech translation. Finally, another future extension could be a semi-supervised approach that would allow to train a system without parallel corpus between all languages in the system.

Bibliography

[Aje18] R. Ajenjo. „Visualizacion de representaciones intermedias en traducciones realizadas por redes neuronales“. Bachelor's Thesis. 2018 (cit. on p. 28).

[Arn94] Doug Arnold. *Machine translation: an introductory guide*. Blackwell Pub, 1994 (cit. on p. 11).

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. „Neural machine translation by jointly learning to align and translate“. In: *arXiv preprint arXiv:1409.0473* (2014) (cit. on pp. 5, 11).

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. „Layer normalization“. In: *arXiv preprint arXiv:1607.06450* (2016) (cit. on p. 6).

[Geh+17] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. „Convolutional Sequence to Sequence Learning“. In: *CoRR* abs/1705.03122 (2017). arXiv: 1705.03122 (cit. on p. 12).

[Gel+17] Spandana Gella, Rico Sennrich, Frank Keller, and Mirella Lapata. „Image Pivoting for Learning Multilingual Multimodal Representations“. In: *arXiv preprint arXiv:1707.07601* (2017) (cit. on p. 13).

[Hot35] H Hotelling. „Canonical correlation analysis (cca)“. In: *Journal of Educational Psychology* (1935) (cit. on p. 10).

[Joh+16] Melvin Johnson, Mike Schuster, Quoc V Le, et al. „Google's multilingual neural machine translation system: enabling zero-shot translation“. In: *arXiv preprint arXiv:1611.04558* (2016) (cit. on p. 12).

[KB14] Diederik P. Kingma and Jimmy Ba. „Adam: A Method for Stochastic Optimization“. In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980 (cit. on p. 19).

[KB18] Lukasz Kaiser and Samy Bengio. „Discrete Autoencoders for Sequence Models“. In: *CoRR* abs/1801.09797 (2018). arXiv: 1801.09797 (cit. on pp. 16, 25).

[KF15] Andrej Karpathy and Li Fei-Fei. „Deep visual-semantic alignments for generating image descriptions“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3128–3137 (cit. on p. 13).

[Koe+07] Philipp Koehn, Hieu Hoang, Alexandra Birch, et al. „Moses: Open source toolkit for statistical machine translation“. In: *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics. 2007, pp. 177–180 (cit. on pp. 11, 22).

[KW13] Diederik P Kingma and Max Welling. „Auto-encoding variational bayes“. In: *arXiv preprint arXiv:1312.6114* (2013) (cit. on p. 8).

[LCB10] Yann LeCun, Corinna Cortes, and CJ Burges. „MNIST handwritten digit database“. In: *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* 2 (2010) (cit. on p. 18).

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. „Effective Approaches to Attention-based Neural Machine Translation“. In: *CoRR* abs/1508.04025 (2015). arXiv: 1508.04025 (cit. on p. 11).

[Lu+18] Yichao Lu, Phillip Keung, Faisal Ladhak, et al. „A neural interlingua for multilingual machine translation“. In: *arXiv preprint arXiv:1804.08198* (2018) (cit. on p. 12).

[MH18] L. McInnes and J. Healy. „UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction“. In: *ArXiv e-prints* (Feb. 2018). arXiv: 1802.03426 [stat.ML] (cit. on p. 28).

[Ng11] Andrew Ng. *CS294A Lecture notes*. Feb. 2011 (cit. on p. 8).

[O+17] Aaron van den Oord, Oriol Vinyals, et al. „Neural discrete representation learning“. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6309–6318 (cit. on p. 9).

[Pap+02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. „BLEU: a method for automatic evaluation of machine translation“. In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2002, pp. 311–318 (cit. on p. 21).

[Sch+17] Holger Schwenk, Ke Tran, Orhan Firat, and Matthijs Douze. „Learning joint multilingual sentence representations with neural machine translation“. In: *arXiv preprint arXiv:1704.04154* (2017) (cit. on p. 12).

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. „Attention is all you need“. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6000–6010 (cit. on pp. 4, 7, 12, 14, 15, 23).

[Vin+08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. „Extracting and composing robust features with denoising autoencoders“. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 1096–1103 (cit. on p. 8).

[Vin+15] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. „Show and tell: A neural image caption generator“. In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE. 2015, pp. 3156–3164 (cit. on p. 13).

List of Figures

List of Tables